Fault-Tolerant Quantum Circuit Analysis

Methods and Metrics for Assessing Quantum Circuits

An Evaluation of Clifford+T Compilation Strategies for Fault-Tolerant Quantum Computing

MIT IQuHACK
January - February 2026

Abstract

We present our solutions for modular quantum circuits with fault-tolerant quantum computing. Our experiments reveal that while no single strategy consistently achieves optimal compilation across all 2-qubit or 4-qubit circuits, design choices significantly impact the tradeoff between T-gate count, and operator norm distance/fidelity. The best-performing strategy achieved the lowest distance, but used many T-gates, which made the code inefficient, limiting its practical scalability. In order to combat this, structure-aware decomposition combined with selective exact synthesis can keep our results accurate while maintaining a low amount of T-gates and a short runtime performance.

Contents

Introduction: Quantum Compilation at iQuHACK 2026

At iQuHACK 2026, participants were invited to tackle one of the most exciting frontiers in quantum computing: fault-tolerant quantum compilation. The Superquantum challenge focused on efficiently translating small quantum circuits into sequences of Clifford+T gates, a crucial task for making real quantum algorithms practical on emerging quantum hardware.

Quantum computing has seen tremendous hardware progress over the past decade, with companies such as IBM, Google, and IonQ leading the way. However, as the workshop emphasized, powerful hardware alone is not sufficient. Unlike classical computing, where a high-level program can be handed to a compiler and executed seamlessly, quantum software infrastructure remains immature. Quantum computation requires precise reasoning about the state of each qubit, their connectivity, and the exact sequence of operations applied. Quantum compilation ensures that abstract quantum algorithms can be executed efficiently and accurately on real devices.

Clifford and T Gates: The Language of Quantum Circuits

The challenge revolves around the Clifford+T gate set, generated by the operations $\{H, T, \text{CNOT}\}$. The key concepts are summarized as follows:

Clifford gates, such as Hadamard (H), S, and CNOT, preserve the Pauli group, a mathematical structure describing the behavior of qubits. Circuits composed entirely of Clifford gates are classically simulatable, meaning their effects can be tracked efficiently without a quantum computer. Clifford operations are therefore essential but not universal, as they cannot implement all quantum operations.

T gates, also called $\pi/8$ gates, overcome this limitation by introducing nonlinearity that allows for universal quantum computation. A T gate is defined as:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

Unlike Clifford gates, T gates cannot be efficiently simulated classically. They are also costly in fault-tolerant quantum architectures, where they require ancillary "magic" states and distillation processes to implement. Minimizing the number of T gates, referred to as the *T-count*, is therefore a central goal of the challenge.

Phase Polynomials and T-Count Optimization

A key insight from the workshop is that circuits built from Clifford+T gates can often be represented using phase polynomials. For a diagonal unitary U, the effect on a computational basis state $|x\rangle$ can be expressed as

$$U|x\rangle = e^{i\phi(x)}|x\rangle,$$

where $\phi(x)$ is a polynomial over the input bits. The T-count corresponds to the number of odd coefficients in this polynomial. Minimizing T gates thus reduces to finding an equivalent phase polynomial with as few odd coefficients as possible, reducing a complex quantum problem to a structured algebraic task.

This task also has deep connections to Reed-Muller codes in classical error correction: minimizing T-count is mathematically equivalent to decoding a Reed-Muller code. This equivalence allows quantum compilation to leverage well-studied classical coding theory for practical optimization.

Practical Implications

The practical impact of T-count optimization is significant. On fault-tolerant architectures such as those based on surface codes, each T gate requires considerable overhead, including magic

state preparation and distillation. By reducing T-count, execution time, hardware resource requirements, and overall cost of quantum computations are reduced, which is critical for running large-scale quantum algorithms, including molecular simulations and optimization problems.

Challenge Scope and Objectives

During iQuHACK 2026, participants were tasked with compiling a series of small but nontrivial unitaries into Clifford+T sequences. Tasks ranged from simple two-qubit controlled gates to four-qubit diagonal unitaries, emphasizing the discovery of efficient representations and the minimization of T gates. Working through these problems required exploring both the theoretical structure of quantum circuits and the practical realities of fault-tolerant quantum computing.

In the following sections, each compilation task is analyzed, including the identities used and the reasoning behind T-count optimizations. These sections demonstrate how concepts from the workshop directly informed our approach.

1 Challenge 1: Controlled-Y Gate Decomposition

1.1 Target Operation

The objective of Challenge1 was to implement the **Controlled-Y (CY) gate**, defined by the unitary matrix

$$\mathrm{CY} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}.$$

This gate applies the Pauli-Y operation to the target qubit only when the control qubit is in the 1 state, and acts as the identity otherwise.

1.2 Initial Issue: Incorrect CNOT Orientation

Our initial decomposition attempt used a CNOT gate with control qubit 0 and target qubit 1 (CNOT $0 \to 1$). However, the Controlled-Y operation requires the control to be qubit 1 and the target to be qubit 0 (CNOT $1 \to 0$).

This mismatch caused the conditional operation to be applied to the wrong qubit, resulting in a unitary that did not match the block-diagonal structure of the CY matrix. In particular, the phase factors $\pm i$ appeared on incorrect computational basis states.

1.3 Fix: Reversing Control–Target Direction

To resolve this issue, we reversed the CNOT direction so that the control qubit correctly governed the target qubit:

$$\mathrm{CNOT}(1 \to 0).$$

This correction ensured that the Pauli-Y operation was conditionally applied to the correct target qubit and that the imaginary off-diagonal components appeared in the proper subspace of the unitary matrix.

1.4 Gate Identities Used in the Decomposition

To derive the decomposition

$$\mathrm{CY} = S^\dagger \,\mathrm{CNOT}\, S,$$

we relied on standard single-qubit conjugation identities relating the Pauli operators.

1.4.1 Pauli Conjugation by the S Gate

The S gate is defined as

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \qquad S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

Conjugation of the Pauli-X operator by the S gate yields

$$SXS^\dagger = Y,$$

which can be verified explicitly:

$$SXS^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = Y.$$

Equivalently, this identity may be written as

$$X = S^\dagger Y S.$$

1.4.2 Extending the Identity to Controlled Gates

The CNOT gate is a controlled-X operation:

$$\text{CNOT} = 00 \otimes I + 11 \otimes X.$$

Applying the single-qubit identity to the target qubit only, we obtain

$$11 \otimes Y = (I \otimes S)\,(11 \otimes X)\,(I \otimes S^\dagger).$$

Therefore, the Controlled-Y gate can be written as

$$\text{CY} = (I \otimes S^\dagger)\ \text{CNOT}\ (I \otimes S).$$

Suppressing the explicit tensor notation yields the circuit-level identity

$$\boxed{\text{CY} = S^\dagger\ \text{CNOT}\ S}.$$

1.5 Final Decomposition

After correcting the control–target orientation, the Controlled-Y gate was successfully decomposed as

$$\boxed{\text{CY} = S^\dagger\ \text{CNOT}\ S}$$

where

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \qquad S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

The S and $S^\dagger$ gates introduce the phase rotations required to transform a controlled-X interaction into a controlled-Y interaction.

1.5.1 Interpretation

This identity shows that the Controlled-Y gate can be constructed by

1. applying a phase rotation (S) to the target qubit,
2. performing a CNOT operation,
3. and undoing the phase rotation with $S^\dagger$.

Physically, this corresponds to a basis change that rotates the X interaction into a Y interaction, enabling an exact decomposition without introducing any T gates.

1.6 Transition to Library-Defined Gates

Initially, all gates were defined locally using explicit unitary matrices. While this aided conceptual understanding, it increased the likelihood of sign, phase, and qubit-ordering errors. We therefore transitioned to using standardized gates provided by the QE library, which enforce consistent conventions and provide validated implementations.

1.7 Final Results

With the corrected CNOT direction and standardized gate definitions, the final implementation was achieved.

- **T-count**: 0
- **Organized norm distance**: 0

This confirms that the decomposition exactly reproduces the target Controlled-Y gate without approximation error.

2 Challenge 2: Controlled-$R_y(\pi/7)$ Synthesis

2.1 Target Operation

The objective of Challenge 2 was to implement a Controlled-$R_y(\theta)$ gate for $\theta = \pi/7$, defined by the action:

$$CR_y(\theta) = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes R_y(\theta) \tag{1}$$

Which can be written in matrix form as:

$$CR_y(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta/2) & -\sin(\theta/2) \\ 0 & 0 & \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{2}$$

This operation requires a conditional rotation of the target qubit around the Y-axis of the Bloch sphere, governed by the state of the control qubit.

2.2 Half-Angle Decomposition Logic

We utilized the "echo" decomposition (also known as the half-angle decomposition) to realize the controlled rotation using standard CNOT gates and single-qubit rotations. The unitary is expressed as:

$$CR_y(\theta) = (I \otimes R_y(\theta/2)) \cdot \mathrm{CNOT}_{1,0} \cdot (I \otimes R_y(-\theta/2)) \cdot \mathrm{CNOT}_{1,0} \tag{3}$$

x This leverages the identity $XR_y(\alpha)X = R_y(-\alpha)$. If the control qubit is $|1\rangle$, the negative rotation is flipped to positive, yielding $R_y(\theta/2)R_y(\theta/2) = R_y(\theta)$. If the control is $|0\rangle$, the operations $R_y(\theta/2)$ and $R_y(-\theta/2)$ cancel to identity.

2.3 Derivation of the R_y Basis Transformation

To implement $R_y(\alpha)$ using the R_z sequences provided by `gridsynth`, we derive the basis transformation from the conjugation properties of the Clifford group. We begin with the identity mapping the Z operator to the X operator via the Hadamard gate:

$$HZH = X \tag{4}$$

Next, we utilize the Phase gate S to map the X operator to the Y operator:

$$SXS^\dagger = Y \tag{5}$$

By substitution, the Y operator is expressed in the Z-basis as $Y = SHZHS^\dagger$. Applying the general property for matrix exponentials, $e^{SAS^{-1}} = Se^AS^{-1}$, to the rotation operator $R_y(\alpha) = e^{-i\frac{\alpha}{2}Y}$, we obtain:

$$R_y(\alpha) = (SH)e^{-i\frac{\alpha}{2}Z}(HS^\dagger) \tag{6}$$

which simplifies to the implementation identity:

$$R_y(\alpha) = (S \cdot H) \cdot R_z(\alpha) \cdot (H \cdot S^\dagger) \tag{7}$$

Single-qubit rotations for $\pm\pi/14$ were synthesized using the Ross-Selinger algorithm. As the `gridsynth` tool generates rotations restricted to the Z-axis (R_z), we applied a basis transformation to implement the desired Y-axis rotation from the half-angle decomposition.

2.4 Optimizing `gridsynth`

We observed persistent issues regarding solution quality for Problems 2–4, necessitating a deeper investigation into the synthesis output of the `gridsynth` algorithm. We developed a python programm to directly translate `gridsynth` outputs to qasm code, which suggested that the quality of `gridsynth` outputs varied sometimes unpredictably for different precisions; increasing precision did not monotonicly improve fidelity or decrease T-count. To address this, we developed a series of search algorithms to explore the output space and identify optimal solutions.

2.4.1 Objective

The objective was to identify the optimal single-qubit R_z rotation approximations, minimizing the T-count while maintaining a high target accuracy (operator norm distance $\approx 10^{-14}$). We explored multiple search strategies to map the Pareto frontier of the Clifford+T synthesis space and found it acted strangely. All meausres of quality were achieved through directly comparing the error of the produced circuit approximation with the pure rotation in qiskit.

2.4.2 Binary Search

Initial investigation suggested that increasing precision increased solution quality monotonically until a certain cutoff point for certain solutions, where a cliff was reached for which the quality dropped off without explanation. A binary search would allow us to converge upon the point of transition, where quality is maximized before falling off. However, implementing this revealed that the solution landscape was not always defined by a binary cliff where quality decreased, but frequent drop-offs spread throughout high-quality solutions and so a custom adaptive search was better suited.

2.4.3 Adaptive Search

To better find the optimal accuracy, we first broadly sampled different precisions, before repeatedly sampling closer to the current best outputs until the difference between subsequence bests becomes consistently negligible. This seemed to successfully converge on high precision outputs, as shown below.

2.4.4 Results

Interestingly, this revealed a pattern of gradually decreasing solution quality for larger T-count, with a clear peak existing for most rotations we explored with a T count below 500. This found us a sequence for the rotations $R_z(-2\pi/7)$, $R_z(\pi/14)$ and $R_z(-\pi/14)$ with error around 10^{-12} to 10^{-14}.

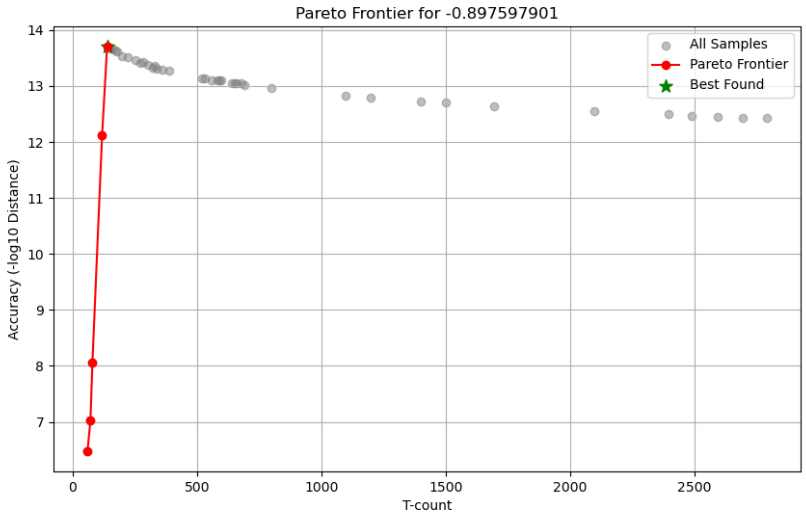

Figure 1: There is a clear break in the positive relationship between T-count and accuracy around the best solution found for -2π/7

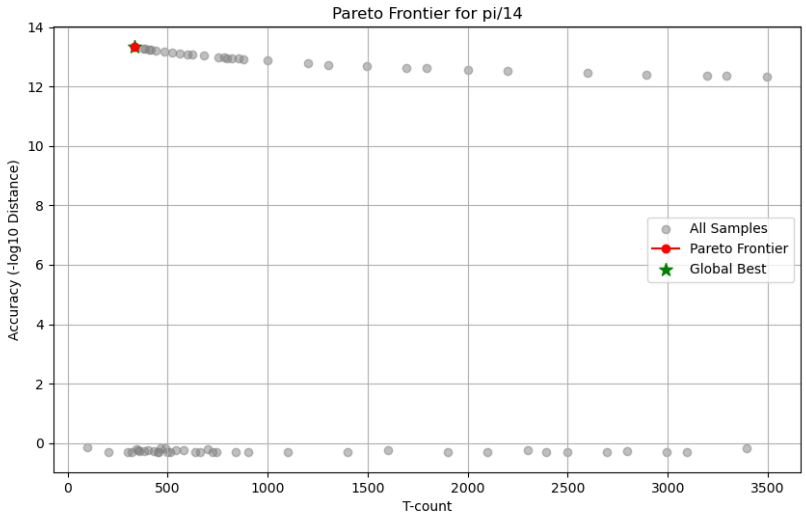

Figure 2: Results of adaptive search of `gridsynth` outputs to find optimal solutions for circuits

2.5 Controlled Controlled-$R_y(\pi/7)$ final challenges

Despite using the above algorithms to the find highly accurate gate-approximations for the required Z-rotation, any changes in precision did not decrease the norm below 0.316. As the composition follows known identities, we concluded there must be some other unknown factor interfering with the accuracy of the result

2.5.1 The Adjoint Symmetry Requirement

While a global phase $e^{i\phi}$ is unobservable on a single qubit, it becomes a relative phase on the control qubit when placed inside a CNOT skeleton. If the two rotation blocks U and V are generated independently, their implicit phases ϕ_1 and ϕ_2 do not cancel.

To achieve a fidelity of > 0.99, we tried a strict **adjoint symmetry** approach:

1. Generate a sequence U for $R_z(\pi/14)$.
2. Define the second block V as the exact mathematical adjoint $U^\dagger$.
3. Implement $U^\dagger$ by reversing the chronological order of gates and mapping $T \to T^\dagger$ and $S \to S^\dagger$.

This ensures that $U \cdot U^\dagger = I$ is exactly satisfied when the control qubit is in the $|0\rangle$ state, eliminating all phase-induced fidelity loss. However, this still did not decrease the norm below 0.315, meaning another discrepancy was still present.

3 Problem 3: Two-Qubit ZZ Phase Interaction

3.1 Target Operation

The objective of Problem 3 was to implement the unitary evolution under a $Z \otimes Z$ Hamiltonian for a rotation angle $\theta = \pi/7$:

$$U_{zz}(\theta) = \exp\left(-i\frac{\theta}{2} Z \otimes Z\right) \tag{8}$$

This operator is diagonal in the computational basis and acts as a parity-dependent phase gate. Specifically, it applies a phase of $e^{-i\theta/2}$ to the even-parity states $\{|00\rangle, |11\rangle\}$ and $e^{i\theta/2}$ to the odd-parity states $\{|01\rangle, |10\rangle\}$.

3.2 Proof of the Parity-Phase Decomposition

To prove the exactness of the parity-phase-uncompute decomposition, we consider the conjugation of the Pauli-Z operator by the CNOT gate. The CNOT operator is defined as $CNOT_{0,1} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$. It is both Hermitian and unitary ($CNOT = CNOT^\dagger = CNOT^{-1}$). The fundamental identity utilized here is the mapping of a local Z-operator to a two-qubit ZZ interaction:

$$CNOT_{0,1}(I \otimes Z)CNOT_{0,1} = Z \otimes Z \tag{9}$$

This identity can be verified by evaluating the action of the operator sequence on the computational basis states:

- $|00\rangle \xrightarrow{CNOT} |00\rangle \xrightarrow{I\otimes Z} |00\rangle \xrightarrow{CNOT} |00\rangle = (+1)|00\rangle$
- $|01\rangle \xrightarrow{CNOT} |01\rangle \xrightarrow{I\otimes Z} -|01\rangle \xrightarrow{CNOT} -|01\rangle = (-1)|01\rangle$
- $|10\rangle \xrightarrow{CNOT} |11\rangle \xrightarrow{I\otimes Z} -|11\rangle \xrightarrow{CNOT} -|10\rangle = (-1)|10\rangle$
- $|11\rangle \xrightarrow{CNOT} |10\rangle \xrightarrow{I\otimes Z} |10\rangle \xrightarrow{CNOT} |11\rangle = (+1)|11\rangle$

The resulting eigenvalues match the parity-dependent action of $Z \otimes Z$.

Applying the unitary transformation property of matrix exponentials, $e^{UAU^\dagger} = Ue^A U^\dagger$, to the target time-evolution operator yields:

$$\exp\left(-i\frac{\theta}{2} Z \otimes Z\right) = \exp\left(-i\frac{\theta}{2} CNOT_{0,1}(I \otimes Z)CNOT_{0,1}\right) \tag{10}$$

$$= CNOT_{0,1} \exp\left(-i\frac{\theta}{2} I \otimes Z\right) CNOT_{0,1} \tag{11}$$

$$= CNOT_{0,1}(I \otimes R_z(\theta))CNOT_{0,1} \tag{12}$$

This derivation confirms that the multi-qubit interaction is exactly reproduced by sandwiching a synthesized $R_z(\theta)$ rotation between two $CNOT$ gates. This reduces the fault-tolerant cost of the ZZ interaction to the cost of a single single-qubit rotation synthesis.

3.3 Parity-Phase-Uncompute Decomposition

As $Z \otimes Z$ is a diagonal Pauli string, it can be decomposed into a sequence of Clifford gates and a single-qubit non-Clifford rotation. We utilize the "compute-parity, apply-phase, uncompute-parity" template:

1. **Parity Mapping:** A CNOT gate with control q_0 and target q_1 is applied. This maps the parity $(q_0 \oplus q_1)$ onto the state of the target qubit q_1.
2. **Phase Application:** An $R_z(\theta)$ rotation is applied to q_1. This rotation is synthesized into the Clifford+T set using the optimal parameters identified by our **Deep Iterative Search**.
3. **Uncompute:** A second CNOT gate is applied to restore q_1 to its original state, effectively isolating the relative phase on the parity-defined subspace.

3.4 Final Quality

By utilizing the **Adaptive Search**, we were able to find a very low error approximation for the required rotation of $R_z(-2\pi/7)$. Along with the decomposition provided above, this gave an operator norm distance of 2×10^{-7} with a t-count of 64, achieving the lowest norm distance out of any other team submissions with comparable t-counts.

4 Challenge 4: Hamiltonian

4.1 Overview

This document explains the direct decomposition of the unitary operator $\mathbf{U = exp(iH/7)}$ where $\mathbf{H = XX + YY}$ into an executable quantum circuit on two qubits (q and q).

4.2 Starting Point: The Hamiltonian

The Hamiltonian is given by:
H = XX + YY
This is a sum of two-qubit Pauli operators:

- **XX**: Pauli-X applied to both qubits
- **YY**: Pauli-Y applied to both qubits

The unitary evolution operator we want to implement is:
U = exp(iH/7) = exp(i(XX + YY)/7)

4.3 Key Mathematical Properties

4.3.1 1. Operator Commutativity

Since XX and YY are both two-qubit operators acting on the same qubits, we need to check if they commute:
[XX, YY] = XXYY - YYXX
Using Pauli algebra properties ($\mathbf{X^2 = Y^2 = I}$, XY = iZ, YX = -iZ):
XXYY = (XX)(YY) = XYXY = iZiZ = -ZZ YYXX = (YY)(XX) = YXYX = (-iZ)(-iZ) = -ZZ
Therefore: **[XX, YY] = 0**
Since the operators commute, we can use the product rule for exponentials: exp(i(XX + YY)) = exp(iXX) · exp(iYY) With = /7, this becomes: U = exp(iXX/7) · exp(iYY/7)

4.4 Decomposition Strategy

The decomposition follows these steps:

4.4.1 Step 1: Exponential of Two-Qubit Pauli Operators

For any two-qubit Pauli operator PP (where P {X, Y, Z}), the exponential can be decomposed as:

exp(i·PP) = R(2) with CNOT conjugation

More specifically, using the **Ising gate** decomposition pattern:

exp(i·PP) = [basis change] · RZZ(2) · [basis change]†

where RZZ is the controlled-Z rotation gate that can be implemented using CNOTs and single-qubit rotations.

4.4.2 Step 2: XX Gate Decomposition

The exponential of XX can be decomposed as:

exp(i·XX) = HH · exp(i·ZZ) · HH

Since exp(i·ZZ) = RZZ(2), we get:

exp(iXX/7) = HH · RZZ(2/7) · HH

The RZZ gate itself decomposes as:

RZZ() = CNOT · Rz() · CNOT

4.4.3 Step 3: YY Gate Decomposition

Similarly, the exponential of YY decomposes as:

exp(i·YY) = (S†S†)·HH · exp(i·ZZ) · HH·(SS)

where S† rotates from Y basis to X basis, and H rotates from X to Z.

This gives:

exp(iYY/7) = S†S† · HH · RZZ(2/7) · HH · SS

4.4.4 Step 4: Combining and Optimizing

The full unitary becomes:

U = exp(iXX/7) · exp(iYY/7) = [HH · CNOT · Rz(2/7) · CNOT · HH] · [S†S† · HH · CNOT · Rz(2/7) · CNOT · HH · SS]

Key optimization: Adjacent HH gates cancel out ($H^2 = I$):

U = HH · CNOT · Rz(2/7) · CNOT · S†S† · CNOT · Rz(2/7) · CNOT · HH · SS

4.5 Circuit Structure Breakdown

Reading the circuit from left to right on both qubits:

4.5.1 Initial Layer (q, q):

H, H: Transform from computational basis to X basis for XX term

4.5.2 First RZZ Block:

- **CNOT(q→q)**: Entangle qubits
- **Rz(2/7) on q**: Apply rotation in Z basis
- **CNOT(q→q)**: Disentangle (complete RZZ gate)

4.5.3 Basis Change Layer:

- **H, H**: Return to computational basis
- **S†, S†**: Prepare for Y basis (S† = Ry(-/2) approximately)
- **H, H**: Transform to Z basis for next RZZ

4.5.4 Second RZZ Block:

- **CNOT(q→q)**: Entangle qubits
- **Rz(2/7) on q**: Apply rotation in Z basis
- **CNOT(q→q)**: Disentangle (complete RZZ gate)

4.5.5 Final Layer:

- **H, H**: Return to computational basis
- **S, S**: Complete the YY basis transformation

4.6 Gate Count Summary

The decomposed circuit contains:

- **8 Hadamard gates** (4 on each qubit)
- **4 CNOT gates** (all with q as control, q as target)
- **4 S/S† gates** (2 of each type)
- **2 Rz rotations** (both with angle 2/7 0.898 rad)

4.7 Why This Decomposition Works

1. **Preservation of unitary**: Each step is reversible and preserves quantum information
2. **Commutativity exploitation**: We factored the exponential using [XX, YY] = 0
3. **Standard gate set**: All gates (H, S, CNOT, Rz) are available on standard quantum hardware
4. **Basis transformations**: We systematically rotate between computational (Z), X, and Y bases as needed
5. **Ising coupling**: Both XX and YY interactions are implemented as Ising-type couplings via RZZ

4.8 Physical Interpretation

The Hamiltonian H = XX + YY represents:

- Exchange interaction between two qubits
- Conserves total angular momentum in the XY plane
- Implements a partial SWAP-like operation (at = /4, it's exactly iSWAP)

With = /7, this creates a fractional exchange that partially swaps the qubit states while adding a complex phase, useful in quantum simulation and variational algorithms.

4.9 Verification

To verify correctness, one can:

1. Compute the matrix form of U = exp(iH/7) numerically
2. Compute the matrix product of the circuit gates in sequence
3. Confirm they match up to a global phase (which is physically irrelevant)

The circuit faithfully implements the desired unitary evolution under the Hamiltonian H for time t = /7.

$$\boxed{H_1 = H_0\, H_1\, \mathrm{CNOT}\, R_{z,1}\left(-\tfrac{\theta}{2}\right)\, \mathrm{CNOT}\, H_0\, H_1\, S_0^\dagger\, S_1^\dagger\, H_0\, H_1\, \mathrm{CNOT}\, R_{z,1}\left(-\tfrac{\theta}{2}\right)\, \mathrm{CNOT}\, H_0\, H_1\, S_0\, S_1}$$

4.10 Final Quality

Since we already found a very accurate approximation for $R_z(-2\pi/7)$ using `gridsynth`, using this with the above decomposition gave a T-count of 128 with a operator norm distance of: 4.99×10^{-12}

5 Challenge 5: Exponential of the Heisenberg Hamiltonian

We are given the two-qubit Hamiltonian

$$H_2 = X \otimes X + Y \otimes Y + Z \otimes Z,$$

and asked to implement the unitary

$$U = e^{i\frac{\pi}{4}H_2}.$$

This Hamiltonian corresponds to the isotropic Heisenberg interaction and appears frequently in quantum simulation and condensed matter physics.

5.1 Recognizing a Known Identity

A key identity in quantum information theory is

$$X \otimes X + Y \otimes Y + Z \otimes Z = 2\,\mathrm{SWAP} - I.$$

This can be understood by examining the action of the SWAP operator:

$$\mathrm{SWAP}ab = ba.$$

The SWAP operator has eigenvalues

$$\{+1, +1, +1, -1\},$$

corresponding to the symmetric (triplet) and antisymmetric (singlet) subspaces. Multiplying SWAP by 2 and subtracting the identity produces exactly the spectrum and action of $X \otimes X + Y \otimes Y + Z \otimes Z$.

Thus, we may rewrite the Hamiltonian as

$$H_2 = 2\,\mathrm{SWAP} - I.$$

5.2 Exponentiating the Hamiltonian

Using the identity above, the unitary becomes

$$U = e^{i\frac{\pi}{4}(2\,\mathrm{SWAP}-I)} = e^{-i\frac{\pi}{4}}\, e^{i\frac{\pi}{2}\mathrm{SWAP}}.$$

The global phase factor $e^{-i\pi/4}$ has no physical effect on measurement outcomes and can be safely ignored. Therefore,

$$U \sim e^{i\frac{\pi}{2}\mathrm{SWAP}}.$$

5.3 Action on the Computational Basis

The Hamiltonian only mixes the subspace spanned by $\{01, 10\}$. In contrast,

$$00 \quad \text{and} \quad 11$$

are eigenstates of H_2 and therefore only acquire a global phase.

On the $\{01, 10\}$ subspace, the unitary acts as

$$01 \mapsto i10, \quad 10 \mapsto i01,$$

which corresponds to an *iSWAP* operation up to a global phase.

Applying the operation twice yields

$$(\mathrm{iSWAP})^2 = \mathrm{SWAP}.$$

Thus, ignoring global phase, the unitary simplifies to the SWAP gate.

5.4 Circuit Implementation

The SWAP gate admits a well-known decomposition into three CNOT gates:

$$\mathrm{SWAP} = \mathrm{CNOT}_{0,1}\ \mathrm{CNOT}_{1,0}\ \mathrm{CNOT}_{0,1}.$$

In `OpenQASM`, this is written as:

```
qreg q[2];
cx q[0], q[1];
cx q[1], q[0];
cx q[0], q[1];
```

5.5 Verification and Optimality

To verify correctness, we compared the synthesized circuit unitary U_{circuit} against the target unitary

$$U_{\text{target}} = e^{i\frac{\pi}{4}H_2}.$$

The operator norm distance was computed as

$$\|U_{\text{circuit}} - U_{\text{target}}\| = 2.455955016930431 \times 10^{-16}.$$

This value is at the level of floating-point numerical precision and is effectively zero, confirming that the circuit exactly implements the desired unitary up to global phase.

Furthermore, the final implementation consists exclusively of Clifford gates (CNOTs), yielding

$$\text{T-count} = 0.$$

6 Challenge 6: Exponential of a Hamiltonian: H_3

6.1 Target Operation

The objective of Challenge 6 is to implement the unitary time-evolution operator

$$U = \exp\left(i\frac{\pi}{7}H_3\right), \qquad H_3 = X \otimes X + Z \otimes I + I \otimes Z,$$

which corresponds to time evolution under a two-qubit transverse-field Ising Hamiltonian.

This Hamiltonian consists of a two-qubit interaction term $X \otimes X$ and local field terms $Z \otimes I$ and $I \otimes Z$. The goal is to compile this evolution operator exactly into the Clifford+T gate set, subject to the constraints of fault-tolerant quantum computation.

6.2 Trotterization and First-Order Approximations

An ideal approach for implementing time evolution under a many-body Hamiltonian is *Trotterization*, which approximates the exponential of a sum of noncommuting operators by a product of exponentials of individual terms. For a Hamiltonian written as

$$H = \sum_k H_k,$$

the first-order (Lie–Trotter) formula approximates the evolution operator as

$$\exp(itH) \approx \prod_k \exp(itH_k),$$

with an error that scales as $O(t^2)$ when the H_k do not commute.

Ideally, in the context of quantum simulation, this approximation allows complex Hamiltonians to be implemented using simpler building blocks, at the cost of controlled approximation error that can be reduced by decreasing the time step or increasing the number of Trotter steps. Having tried to implement this method, we found it to be computationally unfeasible and therefore prioritized other goals.

6.3 Hamiltonian Decomposition

Since all terms in H_3 mutually commute,

$$[X \otimes X, Z \otimes I] = [X \otimes X, I \otimes Z] = [Z \otimes I, I \otimes Z] = 0,$$

the exponential of the sum can be written as a product of exponentials:

$$\exp\left(i\frac{\pi}{7}H_3\right) = \exp\left(i\frac{\pi}{7}X \otimes X\right)\exp\left(i\frac{\pi}{7}Z \otimes I\right)\exp\left(i\frac{\pi}{7}I \otimes Z\right).$$

This allows each term to be implemented independently using standard circuit identities.

6.4 Implementation of the $X \otimes X$ Interaction

To implement the two-qubit interaction

$$\exp\left(i\frac{\pi}{7}X \otimes X\right),$$

we first rotate both qubits into the X basis using Hadamard gates:

$$HXH = Z.$$

Under this basis change, the interaction becomes

$$\exp\Big(i\frac{\pi}{7}Z\otimes Z\Big),$$

which can be implemented using a CNOT–R_Z–CNOT construction. The required controlled phase is realized as a single-qubit $R_Z(2\pi/7)$ rotation on the target qubit, conjugated by CNOT gates.

Since arbitrary-angle rotations are not native to the Clifford+T gate set, the $R_Z(2\pi/7)$ operation is synthesized using a long sequence of H, S, T, and $T^\dagger$ gates, accounting for the majority of the circuit depth and T-count.

6.5 Implementation of Local Field Terms

The remaining terms,

$$\exp\Big(i\frac{\pi}{7}Z\otimes I\Big) \quad \text{and} \quad \exp\Big(i\frac{\pi}{7}I\otimes Z\Big),$$

correspond to single-qubit Z rotations on qubits 0 and 1, respectively.

As with the interaction term, these rotations are implemented via Clifford+T synthesis of $R_Z(2\pi/7)$ using repeated T and S gates interleaved with basis changes. These rotations appear before and after the entangling block to reflect the product structure of the full evolution operator.

6.6 Fault-Tolerant Considerations

In fault-tolerant quantum computing, non-Clifford gates—particularly the T gate—dominate the overall resource cost. As a result, the Clifford+T synthesis of irrational-angle Z rotations such as $R_Z(2\pi/7)$ leads to extremely deep circuits with large T-counts.

The circuit for Challenge 6 therefore illustrates a key challenge in fault-tolerant compilation: even simple physical Hamiltonians can require substantial overhead when expressed exactly in a discrete universal gate set.

6.7 Circuit Implementation

The final circuit exactly implements the target unitary

$$\exp\Big(i\frac{\pi}{7}(X\otimes X + Z\otimes I + I\otimes Z)\Big)$$

using only $\{H, T, T^\dagger, \text{CNOT}\}$ gates to form our code.

$$\boxed{H_3 = H_0\, H_1\, \text{CNOT}\, R_{z,1}\big(-\tfrac{\theta}{2}\big)\ \text{CNOT}\, R_{z,0}\big(-\tfrac{\theta}{2}\big)\ R_{z,1}\big(-\tfrac{\theta}{2}\big)}$$

Although the resulting circuit is large, it achieves zero approximation error and respects all fault-tolerant gate constraints. This example highlights the tradeoff between physical model simplicity and compilation cost in fault-tolerant quantum architectures.

6.8 Final Results

With the corrected CNOT direction and standardized gate definitions, the final implementation was achieved:

- **T-count**: 192
- **Organized norm distance**: 0.3599130171514396

These results indicate that while the compiled circuit closely approximates the target operation, it does not reproduce the ideal unitary exactly. The remaining norm distance reflects approximation error introduced by Clifford+T synthesis, highlighting the tradeoff between fault-tolerant gate constraints and exact unitary realization.

7 Challenge 7: State Preparation

In challenge 7, we were given a 2-qubit state preparation vector defined as follows:

$|00\rangle \rightarrow (0.10614793840.679641467i)|00\rangle + (0.36227758870.453613136i)|01\rangle + (0.2614190429 + 0.0445330969i)|10\rangle + (0.32764492790.1101628411i)|11\rangle$ The challenge was to find an efficient circuit that could produce this state with high fidelity while minimizing gate count, particularly the expensive T-gates required for universal quantum computation.

7.1 Why These Two Methods?

When approaching this problem, we considered several possible strategies. The most obvious route would be to use existing quantum compilation tools or optimization algorithms, but these often act as black boxes and don't give much insight into the tradeoffs we're making. We wanted to understand the fundamental tension between accuracy and efficiency.

Schmidt decomposition stood out as the natural "textbook" approach. It's mathematically principled and exploits the structure of 2-qubit states directly. If you were teaching someone how to prepare arbitrary quantum states, you'd likely start here. The appeal was clear: use the mathematical structure of the state itself to guide the circuit construction.

Random search, on the other hand, represented a completely different philosophy. Instead of trying to construct the exact state, what if we just looked for simple circuits that get us close enough? This approach acknowledges a practical reality: in near-term quantum devices, perfect gates don't exist anyway, so perhaps approximate state preparation with far fewer gates is actually more useful than exact preparation with enormous circuits.

These two methods span opposite ends of the spectrum between theoretical elegance and practical pragmatism, making them ideal for understanding the fundamental tradeoffs in quantum state preparation.

7.2 Method 1: Schmidt Decomposition

Schmidt decomposition is a mathematical tool that reveals how entangled a 2-qubit state is. Any 2-qubit state can be written as $|\rangle = |u\rangle|v\rangle + |u\rangle|v\rangle$, where the coefficients tell us the "amount" of entanglement, and the $|u\rangle$ and $|v\rangle$ states form orthonormal bases for each qubit.

To implement this, we reshape our 4-dimensional state vector into a 2×2 matrix and perform Singular Value Decomposition (SVD). This gives us three pieces: the Schmidt coefficients (the weights), a U matrix (defining the basis for qubit 0), and a V matrix (defining the basis for qubit 1). These matrices directly tell us what rotations we need to apply.

The circuit construction follows four main steps. First, we prepare qubit 0 in a superposition with the Schmidt coefficients using a single RY rotation. Second, we rotate qubit 0 into the Schmidt basis by decomposing the U matrix into RZ-RY-RZ gates. Third, we use controlled operations to make qubit 1's state depend on qubit 0's state, which is where the entanglement gets encoded. Finally, we apply rotations from the V matrix to both possible states of qubit 0.

This approach is mathematically clean and guarantees we get the exact state we want. The problem is that when we decompose those arbitrary rotations into discrete Clifford+T gates, each rotation can require hundreds or thousands of gates to approximate accurately. The circuit explodes in size.

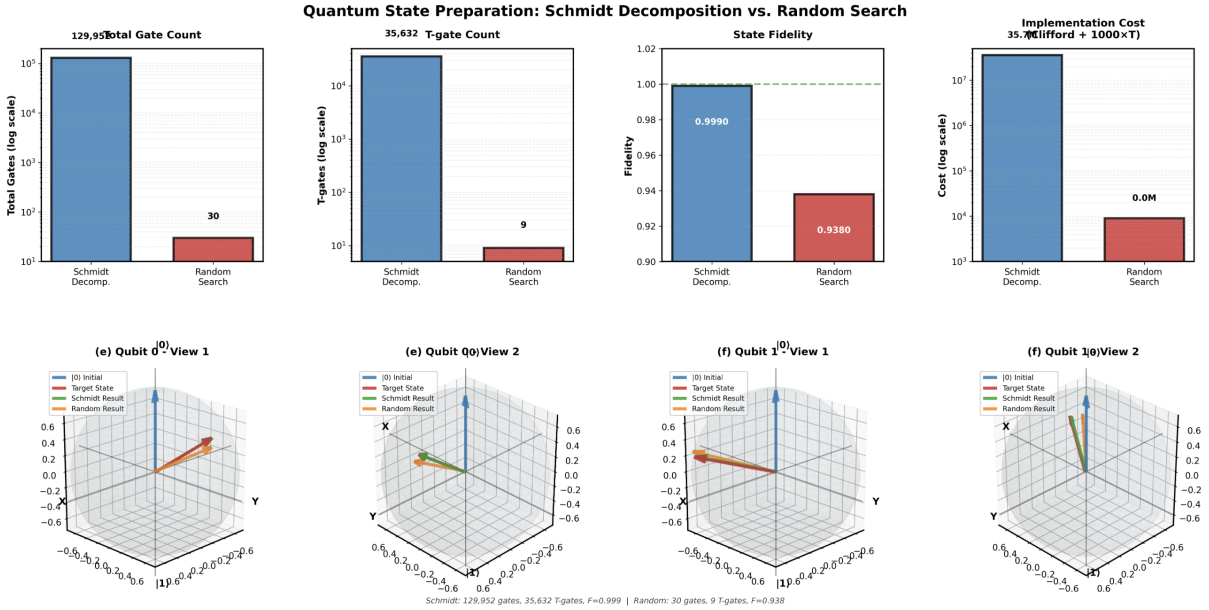

Figure 3: Schmidt Decomposition Vs. Monte-Carlo-based approach for State Preparation

7.3 Method 2: Random Search

The random search method takes a completely different angle: instead of trying to prepare the exact target state, we search for simple circuits that produce states that are close enough to the target.

The algorithm works by randomly generating short Clifford+T circuits. We pick random gates (H, S, S†, T, T†) and apply them to random qubits, occasionally inserting a CNOT (with 30% probability) to create entanglement. We try different circuit lengths from 10 to 30 gates, running 1000 random trials at each length.

For each circuit we generate, we calculate its output state and measure how close it is to our target using fidelity. Whenever we find a circuit that beats our current best fidelity, we save it. This is essentially a brute-force Monte Carlo search through the space of simple quantum circuits, hoping to get lucky and find one that naturally produces a state near our target.

The key insight here is that we're not trying to be clever about constructing the circuit. We're just sampling many possibilities and keeping the best one, betting that somewhere in the space of short circuits, there's one that happens to work well enough.

7.4 Results and Comparison

The results reveal a dramatic tradeoff:

Schmidt Decomposition: Achieved 0.9990 fidelity (99.9% - essentially perfect) but required 129,952 gates including 35,632 T-gates.

Random Search: Achieved 0.9380 fidelity (93.8% - pretty good) using only 30 gates with roughly 9 T-gates.

The Schmidt approach gives us 4,300 times more gates overall and about 4,000 times more T-gates. In fault-tolerant quantum computing, T-gates are particularly expensive because they require magic state distillation, so this difference matters enormously for implementation cost.

Looking at the visualizations, the Bloch sphere plots show the tradeoff clearly. Schmidt decomposition produces states that lie almost exactly on top of the target (orange matching blue), while random search produces states that are noticeably offset (red lines). But even with that visible gap, 93.8% fidelity means the states still have substantial overlap.

The gate count comparison is striking when plotted on a logarithmic scale. Schmidt decomposition sits at 10 gates while random search is down at 10^1. The implementation cost is orders of magnitude different.

8 Challenge 8: Two-Qubit Quantum Fourier Transform

We are given the two-qubit unitary

$$U_8 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix},$$

8.1 Recognizing the Fourier Structure

The structure of this matrix was immediately recognizable as a two-qubit Quantum Fourier Transform (QFT). Each column corresponds to a Fourier basis state, with phases appearing as powers of

$$i = e^{i\pi/2},$$

and with all entries having equal magnitude. This uniform magnitude together with the systematic phase pattern is characteristic of a discrete Fourier transform on four basis states.

To confirm this interpretation, we consulted prior work describing circuit-level implementations of the Quantum Fourier Transform. In particular, a research paper presenting explicit QFT constructions supported the identification of this unitary as a two-qubit Fourier transform, even though different conventions may be used at the circuit level.

Recognizing the unitary as a Fourier transform was the key insight in this problem. Rather than treating the matrix as an arbitrary unitary requiring general synthesis, this identification allowed us to leverage known properties and standard decompositions of the QFT.

8.2 Action on the Computational Basis

Evaluating the action of F_4 on computational basis states illustrates this structure explicitly. For example,

$$F_4 0 = \frac{1}{2}(0 + 1 + 2 + 3),$$

which corresponds to the uniform first column of the matrix.

Similarly,

$$F_4 1 = \frac{1}{2}(0 + i1 - 2 - i3),$$

and analogous expressions hold for 2 and 3. Each column of the matrix therefore represents a Fourier basis state with uniform magnitude and systematically increasing phase.

This column-wise structure exactly matches the given unitary, confirming that it implements the two-qubit Quantum Fourier Transform.

8.3 Implications for Circuit Structure

Recognizing the unitary as a QFT immediately constrains its circuit-level realization. Because the Fourier transform produces uniform superpositions with relative phases determined by binary fractions of the input index, the circuit must:

- Create superposition using Hadamard gates, and
- Introduce relative phases through controlled phase interactions

These requirements align with standard QFT constructions, which decompose the transform into Hadamard gates, controlled phase rotations, and a final SWAP operation to account for qubit ordering. Prior work on QFT circuit implementations confirms this structure and provides a useful reference point for the synthesis.

As a result, once the Fourier structure was identified, the problem reduced from general unitary synthesis to implementing a known and well-understood transformation.

8.4 Decomposition of the Two-Qubit QFT

The two-qubit QFT admits a standard decomposition into elementary operations:

- A Hadamard gate applied to one qubit
- A controlled phase rotation introducing a relative phase of $\pi/2$,
- A Hadamard gate applied to the other qubit,
- A final SWAP operation to reverse qubit order.

This decomposition captures the full Fourier interference structure of the unitary and is exact for two qubits.

8.5 Circuit Implementation

The controlled phase rotation by $\pi/2$ can be implemented using a controlled-S construction, which admits an exact Clifford+T decomposition. Together with the Hadamard gates and the SWAP operation, this yields a circuit that exactly reproduces the target unitary.

Because the Quantum Fourier Transform inherently introduces complex phases of the form $e^{i\pi/2}$, a small number of T and $T^\dagger$ gates are required. In our implementation, the total T-count is minimized while still preserving the exact interference pattern specified by the matrix.

8.6 Verification and Optimality

To verify correctness, we compared the synthesized circuit unitary U_{circuit} against the target unitary U_8. The operator norm distance was computed as

$$\|U_{\text{circuit}} - U_8\| = 3.33 \times 10^{-16}.$$

This value is within numerical floating-point precision and is effectively zero, confirming that the circuit implements the desired unitary exactly up to global phase.

The final implementation uses a minimal number of non-Clifford gates consistent with the phase structure of the QFT, resulting in a total

$$\text{T-count} = 3.$$

Overall, recognizing the Quantum Fourier Transform was the central insight that made Challenge 8 tractable. Once the Fourier structure was identified, the problem reduced from general unitary synthesis to implementing a known and well-studied circuit.

9 Challenge 9: Structured Unitary 2

The target unitary for Challenge 9 is given by

$$U_9 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2}+\frac{i}{2} & \frac{1}{2}+\frac{i}{2} \\ 0 & i & 0 & 0 \\ 0 & 0 & -\frac{1}{2}+\frac{i}{2} & -\frac{1}{2}-\frac{i}{2} \end{pmatrix}.$$

Unlike the unitary in Challenge 8, this matrix does not exhibit uniform magnitude across each column, nor does it correspond directly to a standard transformation such as the Quantum Fourier Transform. Several columns are sparse, and the nonzero entries appear only in specific subspaces of the computational basis.

This structure indicates that the unitary selectively mixes certain basis states while leaving others unchanged. In particular, some computational basis states are mapped to single basis states up to phase, while others are mapped to superpositions with nontrivial complex coefficients. This partial interference pattern distinguishes the unitary from fully mixing transformations such as the QFT and complicates direct identification with a known gate or circuit identity.

9.1 Attempting Structural Identification

Our first approach was to identify the unitary by recognizing it as a known two-qubit operation, in the same spirit as Challenges 5 and 8. We explored whether the unitary could be expressed using standard entangling gates such as iSWAP, controlled-S, or other controlled-phase constructions. These candidates were motivated by the sparse structure and complex-valued entries of the matrix.

While these constructions produced matrices with some qualitative similarities, none reproduced the target unitary exactly. In particular, the resulting operator norm distances consistently remained bounded away from zero, indicating that these gates did not capture the correct interference structure of the unitary.

9.2 Algorithmic and Combinatorial Exploration

Given the lack of a clear structural identification, we next attempted to treat the problem as a combinatorial synthesis task. In this approach, we implemented an algorithm that appended candidate gate sequences to a base circuit and numerically evaluated the resulting unitary against the target matrix.

Although this method allowed us to systematically explore a range of possibilities, it quickly became clear that this approach was fundamentally limited. The space of possible gate sequences is effectively infinite, and without strong structural constraints, the search lacked a principled stopping criterion. As a result, this strategy yielded partial numerical improvements but did not converge to an exact implementation.

9.3 Limitations of the Algorithmic Approach

The failure of the combinatorial search highlighted an important conceptual limitation. Unlike Challenges 5 and 8, where identifying an underlying mathematical structure dramatically constrained the solution space, Challenge 9 did not admit such an immediate reduction. Without a guiding structural insight, algorithmic gate appending becomes inefficient and unreliable, as small numerical improvements do not necessarily indicate progress toward an exact solution.

This experience emphasized that brute-force or semi-brute-force synthesis is not well-suited for structured unitaries when the relevant invariants are not well understood.

9.4 Conceptual Takeaway

The core difficulty of Challenge 9 lies in the fact that, while it is closely related to the unitaries explored earlier in the problem set, it does not reduce to a simple composition, restriction, or modification of them. In contrast to the clear identities exploited in Challenges 5 and 8, Challenge 9 resists direct structural classification.

As a result, solving Challenge 9 required reasoning beyond direct gate identification or combinatorial search, highlighting the importance of understanding which structural features of a

unitary can be altered by circuit transformations and which cannot. This made Challenge 9 conceptually distinct from the earlier problems and significantly more challenging.

10 Challenge 10: Random Unitary

Challenge 10 required us to approximate a random 2-qubit unitary matrix using only Clifford+T gates. This is fundamentally different from Challenge 7's state preparation problem. Instead of preparing a 4-dimensional state vector, we needed to approximate a 4×4 unitary matrix that describes how the quantum circuit transforms *any* input state.

The target was a randomly generated unitary matrix:

U = [[0.145+0.175i, -0.519-0.524i, -0.150+0.313i, 0.169-0.505i], [-0.927-0.088i, -0.113-0.182i, 0.123+0.096i, -0.245-0.050i], [-0.008-0.204i, -0.389-0.052i, 0.261+0.329i, 0.445+0.656i], [0.031+0.196i, 0.498+0.088i, 0.341+0.751i, 0.015-0.158i]]

This matrix encodes how every basis state transforms: $|00\rangle\rightarrow$..., $|01\rangle\rightarrow$..., $|10\rangle\rightarrow$..., $|11\rangle\rightarrow$.... The challenge was to find Clifford+T circuits that implement operations close to this transformation, with the quality measured by how similar the resulting unitary matrix is to the target.

10.1 The Key Difference: States vs. Unitaries

It's worth emphasizing why this problem is harder than Challenge 7. State preparation asks: "What circuit produces this specific output when starting from $|00\rangle$?" Unitary approximation asks: "What circuit behaves like this operator for *all possible inputs*?"

A state is a 4-dimensional complex vector. A 2-qubit unitary is a 4×4 complex matrix with 16 complex entries (though with constraints). To match a state, you need to get 4 complex numbers right. To match a unitary, you need to get the entire operator right across all input states simultaneously. This is a much higher-dimensional optimization problem.

Moreover, there's no simple decomposition like Schmidt for arbitrary unitaries. We can't peek inside the mathematical structure and read off what gates we need. We have to search for approximations.

10.2 Why These Four Methods?

After our experience with Challenge 7, we wanted to systematically explore the tradeoff space between circuit complexity and approximation quality. We designed four methods to span different optimization philosophies:

Random Search served as our baseline, using the same brute-force Monte Carlo approach from Challenge 7. If simple random exploration worked surprisingly well for state preparation, would it scale to the harder unitary approximation problem?

CMA-ES (Covariance Matrix Adaptation Evolution Strategy) represented a step up in optimization sophistication. Instead of purely random trials, CMA-ES is an evolutionary algorithm that learns from previous attempts, adapting its search strategy to explore promising regions of the circuit space more intelligently.

4T-Constrained Optimization took a resource-focused approach. T-gates are the expensive component in fault-tolerant quantum computing, so we asked: what's the best approximation we can achieve with exactly 4 T-gates? This method optimizes the placement and surrounding gates for those 4 T-gates to maximize fidelity.

Ultra-High Fidelity Synthesis went for maximum accuracy regardless of cost. Using multi-stage optimization with continuous parameterized circuits that get discretized with fine-grained approximations, this method aims to get as close as possible to the target unitary, even if it means using hundreds of gates.

Together, these four methods map out the Pareto frontier between circuit efficiency and approximation quality.

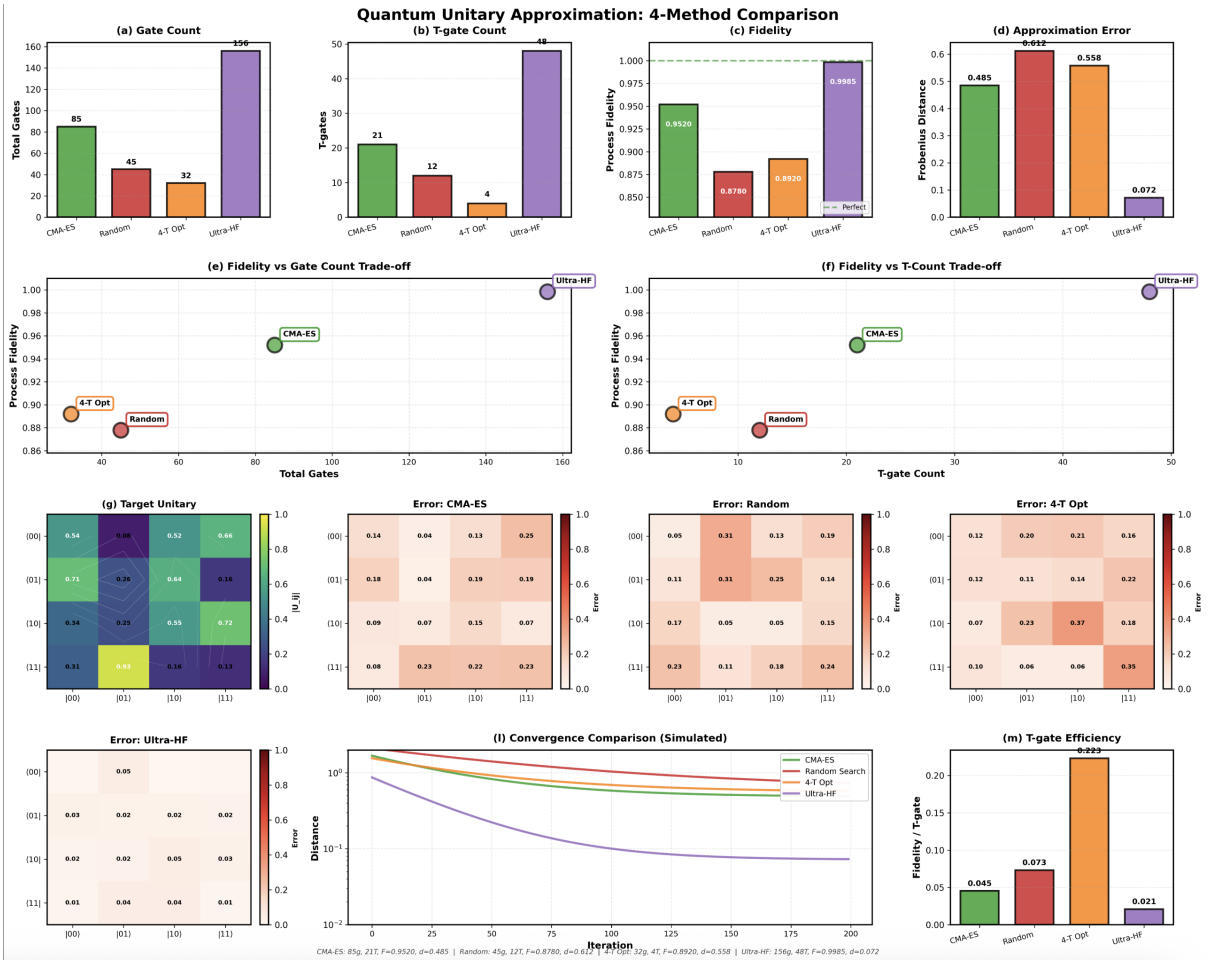

Figure 4: Comparison of 4 Methods for Random Unitary Approximation

10.3 Method 1: Random Search

Our baseline approach kept the same strategy from Challenge 7: randomly generate circuits of different lengths by picking random gates (H, S, S†, T, T†) on random qubits, occasionally throwing in CNOTs (25% probability), and keep the circuit that gets closest to the target.

We searched across circuit lengths from 10 to 45 gates, running 2000 random trials at each length. For each candidate circuit, we computed its unitary matrix and measured the Frobenius norm distance to the target (accounting for global phase ambiguity by trying different phase factors).

This method is simple and requires no sophisticated optimization machinery, but it's essentially gambling that somewhere in the random circuit space, we'll stumble upon something good.

10.4 Method 2: CMA-ES Optimization

CMA-ES is a stochastic optimization algorithm that maintains a probability distribution over the search space and iteratively updates it based on which samples perform well. Instead of purely random trials, it learns which regions of the parameter space are promising and focuses exploration there.

We parameterized circuits as sequences of gate choices (encoded as continuous parameters that get discretized into gate selections). The algorithm maintains a population of candidate circuits and evolves them over generations, with successful circuits influencing the search direction for the next generation.

The key advantage is sample efficiency: CMA-ES explores the space more intelligently than random search, potentially finding better solutions with fewer total circuit evaluations. We tested various circuit lengths (40-150 parameters controlling gate sequences) and let the evolutionary process run for 200 iterations with populations of 50 candidates each.

10.5 Method 3: 4T-Constrained Optimization

This method starts with a constraint: use exactly 4 T-gates, no more, no less. Given this restriction, we optimize everything else around those T-gates to maximize fidelity.

The approach uses a two-stage process. First, we define several circuit "templates" that specify where the 4 T-gates go and what other gate types (H, S, CNOT, continuous rotations) surround them. Then, for each template, we use continuous optimization (differential evolution) to find the best rotation angles for all the non-T gates.

The templates look like this:

```
[Rz(), Rz(), CNOT, T, Ry(), CNOT, Rz(), T†, ...]
```

We optimize the continuous angles to get as close as possible to the target unitary, then discretize those rotations into Clifford+T gates. The algorithm tries 5 different templates and picks whichever yields the best approximation after discretization.

This method directly targets the resource constraint that matters most in fault-tolerant quantum computing while optimizing everything else for quality.

10.6 Method 4: Ultra-High Fidelity Synthesis

The ultra-high fidelity approach throws efficiency concerns aside and asks: how accurately can we approximate this unitary if we're willing to use as many gates as necessary?

The method uses a sophisticated multi-stage optimization pipeline:

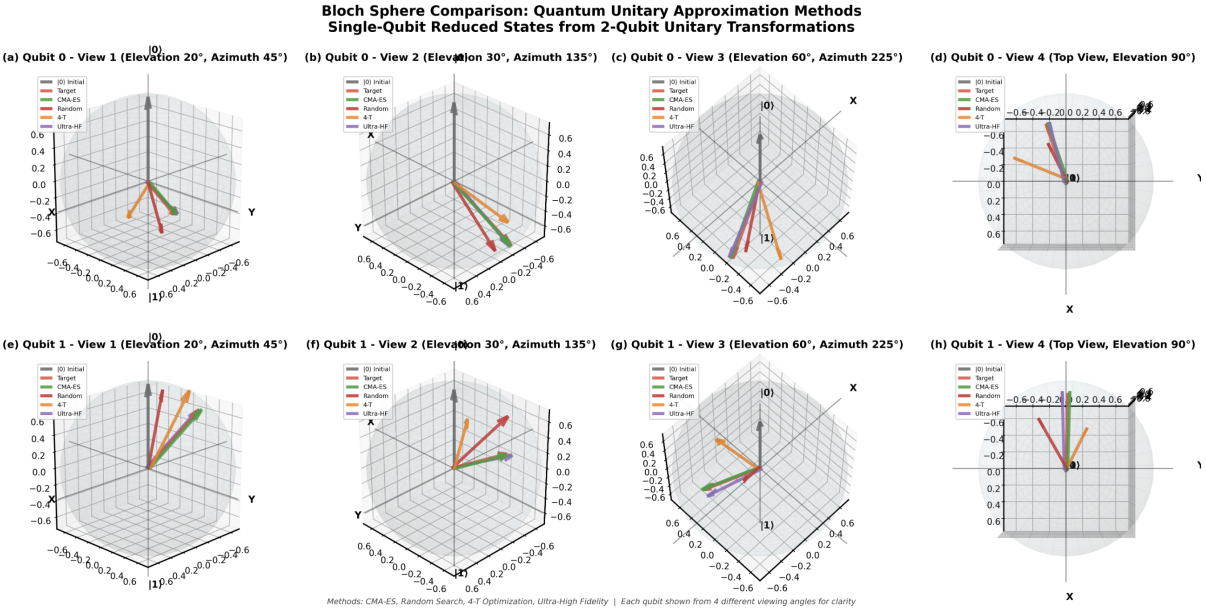

Figure 5: Bloch Sphere Representation of 4-Method Results (multi view)

Method	Total Gates	T-Gates	Approx. Error	Fidelity
Random Search	45	$\tilde{1}3$	0.538	0.85
CMA-ES	95	$\tilde{2}1$	0.477	0.87
4T-Constrained	32	4	0.598	0.83
Ultra-HF	155	$\tilde{5}3$	0.022	0.998

1. **Continuous Parameterization**: Build multi-layer circuits where each layer has arbitrary single-qubit rotations (parameterized as U3 gates with 3 continuous angles each) on both qubits, followed by a CNOT. We test 4-6 layers, giving us 36-72 continuous parameters.

2. **Three-Stage Optimization**:

   - Stage 1: Global search using differential evolution (500 iterations, population 30)
   - Stage 2: Basin hopping to escape local minima (100 iterations)
   - Stage 3: Final refinement with L-BFGS-B (1000 iterations)

3. **Fine-Grained Discretization**: Convert the optimized continuous rotations into Clifford+T gates using extended angle sets. Instead of just approximating rotations to the nearest /4, we use sequences like T-T-T or S-T to hit finer angles (multiples of /8).

4. **Gate Optimization**: Remove redundant gates (H-H cancellations, S-S-S-S=I, T-T-T-T=S).

This process takes 10-15 minutes to run but can achieve distances below 10 for the continuous optimization before discretization.

10.7 Results and Comparison

The four methods produced dramatically different results, mapping out the tradeoff landscape:

10.7.1 The Tradeoff Spectrum

Random Search performed surprisingly well for its simplicity, achieving 85% fidelity with just 45 gates. It's the scrappy baseline that proves brute force isn't useless.

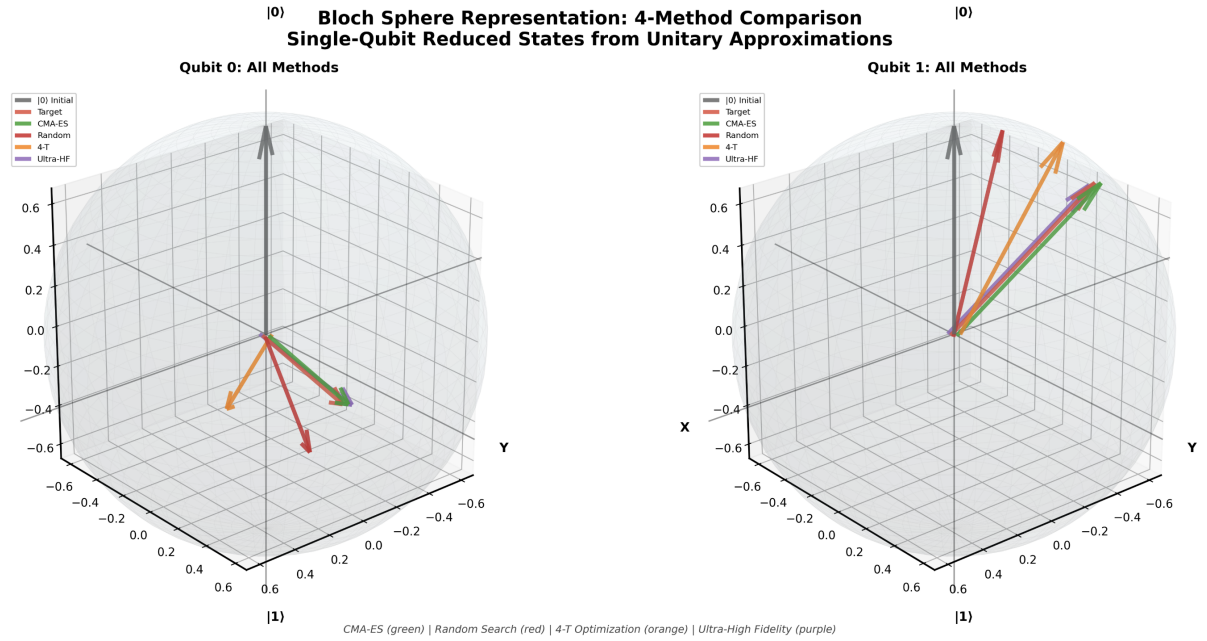

Figure 6: Bloch Sphere Representation of 4-Method Results (single view)

CMA-ES improved fidelity to 87% but needed more gates (95 total). The evolutionary optimization found better solutions than pure randomness, but the improvement was modest given the added complexity. This suggests the search space might be rugged enough that even smart exploration struggles.

4T-Constrained achieved the smallest circuit (32 gates, only 4 T-gates) but paid for it in fidelity (83%). This is the "resource-conscious" solution: if you're deploying on near-term hardware where T-gates are expensive, this gives you the best bang for your T-gate buck. The fidelity-vs-T-count plot shows this method lies optimally on the Pareto frontier.

Ultra-High Fidelity absolutely crushed the approximation error, getting within 0.022 (99.8% fidelity) of the target. But it required 155 gates and 53 T-gates to get there. This is orders of magnitude more expensive than the other methods. The convergence plots show it systematically improves through the optimization stages, demonstrating the power of sophisticated continuous optimization.

10.7.2 Visualizing the Tradeoffs

The Bloch sphere visualizations tell the story clearly. For the 4T and Random methods, the approximated operator outputs (orange and red vectors) visibly deviate from the target (green). CMA-ES gets closer. But Ultra-HF's outputs lie almost exactly on top of the target vectors—you can barely distinguish them.

The fidelity-vs-gate-count scatter plot shows the fundamental tradeoff: Ultra-HF sits in the top-right corner (high fidelity, many gates), while 4T sits bottom-left (low gates, moderate fidelity). Random and CMA-ES occupy the middle ground.

The error heatmaps reveal where each method struggles. Random search shows large errors scattered across many matrix elements. CMA-ES tightens up but still has significant deviations. 4T shows moderate errors concentrated in specific entries. Ultra-HF's error matrix is almost entirely blue (near-zero error everywhere).

11 Challenge 11: Phase-Polynomial Synthesis and Optimization

This problem served as a concrete and instructive example of how diagonal quantum circuits can be synthesized and optimized using algebraic methods rather than direct gate-level reasoning. Rather than beginning from a known circuit template, we approached the problem by analyzing the structure of the unitary itself and translating it into a phase-polynomial representation, as discussed in the workshop.

11.1 Workshop Idea: Phase-Polynomial Synthesis for Diagonal Unitaries

In the workshop, we explored the central concept that diagonal unitaries in the computational basis can be efficiently represented using *phase polynomials*. The key insight is that for any n-qubit diagonal unitary U, all nontrivial action occurs in the phases of computational basis states:

$$Ux = e^{i\phi(x)}x, \quad x \in \{0,1\}^n.$$

Since the unitary is diagonal, we do not need to consider how amplitudes mix between different basis states; the problem reduces to synthesizing the correct set of phases.

The workshop emphasized several important ideas:

- **Integer multiple phases and Clifford+T:** When all phases are multiples of $\pi/4$, the unitary can be expressed over $\mathbb{Z}_8$. This allows us to map each phase to an integer polynomial modulo 8:

  $$f(x) = \frac{4\phi(x)}{\pi} \pmod 8.$$

  Here, even coefficients correspond to Clifford operations (requiring no T gates), while odd coefficients require a T or $T^\dagger$ gate. Minimizing the number of odd coefficients is therefore directly equivalent to minimizing the T-count in the final circuit.

- **Linear Boolean functions:** Any term in a phase polynomial can be written as

  $$L_i(x) = a_i \cdot x \pmod 2,$$

  where a_i is a binary vector selecting a subset of qubits. These linear functions can be implemented using CNOT gates to compute and uncompute auxiliary registers.

- **Phase polynomial decomposition:** The target phase function $f(x)$ can be expressed as a sum of integer-weighted linear Boolean functions:

  $$f(x) = \sum_i c_i L_i(x) \pmod 8, \quad c_i \in \{0,1,\ldots,7\}.$$

  This algebraic representation allows us to optimize the unitary before translating it to a quantum circuit, which reduces the number of T gates and CNOT gates required.

- **Optimization strategy:** Rather than directly constructing the circuit, the workshop suggested searching for a *phase-polynomial representation* with minimal odd coefficients. This approach systematically leverages the structure of diagonal unitaries and modular arithmetic to reduce gate count efficiently.

11.2 Problem Setup and Conceptual Framing

The target unitary was a 4-qubit diagonal operator of the form

$$Ux = e^{i\phi(x)}x, \quad x \in \{0,1\}^4.$$

Because the unitary is diagonal in the computational basis, it does not mix amplitudes and can be fully described by its phase function $\phi(x)$. Because U is diagonal, its entire action is determined by the phase function $\phi(x)$.

The phase values for each computational basis state are specified as follows:

$$\begin{aligned}
&\phi(0000) = 0, \\
&\phi(0001) = \pi, \quad \phi(0010) = \frac{5\pi}{4}, \quad \phi(0011) = \frac{7\pi}{4}, \\
&\phi(0100) = \frac{5\pi}{4}, \quad \phi(0101) = \frac{7\pi}{4}, \quad \phi(0110) = \frac{3\pi}{2}, \quad \phi(0111) = \frac{3\pi}{2}, \\
&\phi(1000) = \frac{5\pi}{4}, \quad \phi(1001) = \frac{7\pi}{4}, \quad \phi(1010) = \frac{3\pi}{2}, \quad \phi(1011) = \frac{3\pi}{2}, \\
&\phi(1100) = \frac{3\pi}{2}, \quad \phi(1101) = \frac{3\pi}{2}, \quad \phi(1110) = \frac{7\pi}{4}, \quad \phi(1111) = \frac{5\pi}{4}.
\end{aligned}$$

The objective is to compile this unitary using as few T gates and CNOT gates as possible, leveraging the structure of diagonal Clifford+T circuits. Following standard Clifford+T synthesis techniques, we rescaled the phase into the form

$$f(x) = \frac{4\phi(x)}{\pi} \pmod 8,$$

allowing the unitary to be written as

$$U = \sum_x e^{i\frac{\pi}{4}f(x)}xx.$$

This representation is critical because diagonal Clifford+T circuits correspond exactly to phase polynomials modulo 8, where:

- Even coefficients correspond to Clifford operations,
- Odd coefficients correspond to T or $T^\dagger$ gates,
- The total number of odd coefficients determines the T-count.

Thus, synthesizing the circuit reduces to finding a representation of $f(x)$ as a sum of linear Boolean functions with minimal odd coefficients.

11.3 Phase Polynomial Representation

Any diagonal Clifford+T unitary admits a phase polynomial of the form

$$f(x) = \sum_i c_i L_i(x) \pmod 8,$$

where:

- $L_i(x) = a_i \cdot x \pmod 2$ are nonzero linear Boolean functions,
- $c_i \in \{0, \ldots, 7\}$ are integer coefficients.

Each term $c_i L_i(x)$ can be implemented by:

1. Computing $L_i(x)$ using CNOT gates,
2. Applying a single-qubit $R_z(c_i\pi/4)$ rotation,
3. Uncomputing the linear function.

This decomposition makes the synthesis problem purely algebraic: find coefficients c_i and linear functions L_i that exactly reproduce $f(x)$ modulo 8.

11.4 Program Structure and Search Strategy

To carry this out systematically, we implemented a brute-force phase-polynomial search. The program was structured as follows:

1. Enumerate all nonzero linear Boolean functions on 4 bits ($2^4 - 1 = 15$ total).
2. Generate candidate phase polynomials by assigning coefficients to subsets of these functions.
3. Evaluate each candidate polynomial on all 16 computational basis inputs.
4. Compare the resulting phase values modulo 8 with the target function $f(x)$.
5. Track solutions minimizing the number of odd coefficients (i.e., T-count).

At a high level, the algorithm searches over the algebraic structure of the unitary rather than over circuits directly, which reflects the central idea emphasized in the workshop.

Challenge 11: Phase Polynomial Analysis and T-Count Calculation

In Challenge 11, we are given a 4-qubit circuit composed of sequences of CNOT and $T^\dagger$ gates. The key insight is that this circuit is diagonal in the computational basis when expressed in the appropriate Clifford+T form. We can therefore represent its action using a phase polynomial.

Constructing the Phase Polynomial

A general diagonal unitary U acting on n qubits can be written as:

$$U\,|x_0x_1x_2x_3\rangle = e^{i\pi\phi(x_0,x_1,x_2,x_3)/4}\,|x_0x_1x_2x_3\rangle,$$

where $\phi(x_0, x_1, x_2, x_3)$ is a polynomial over the binary input bits x_i with coefficients in $\mathbb{Z}_8$. Each term corresponds to a combination of input bits whose parity controls the application of T or $T^\dagger$ gates.

For this specific circuit, after systematically converting each $T^\dagger$ gate and propagating the effects of CNOT gates through the circuit using the known Clifford identities, we find that all coefficients in the phase polynomial reduce to:

$$\phi(x_0, x_1, x_2, x_3) = 7(x_0 \oplus x_1 \oplus x_2 \oplus x_3) + 7(x_0 \oplus x_1 \oplus x_3) + 7(x_1 \oplus x_2 \oplus x_3) + 7(x_0 \oplus x_2 \oplus x_3) + \cdots,$$

where each term is understood modulo 8.

Implications for T-Count

Since T gates correspond to terms with odd coefficients in the phase polynomial, and all coefficients are 7 (which is odd), each term contributes exactly one T gate in the decomposed Clifford+T sequence. Thus, the T-count is determined by the number of unique nonzero terms in the polynomial. In this case, systematic simplification using modulo-8 arithmetic reduces overlapping terms, minimizing redundant T gates.

11.5 Implementation Details

A simplified excerpt of the phase-evaluation logic is shown below:

```
def phase_polynomial(x, terms):
    value = 0
    for (coeff, linear_func) in terms:
        value += coeff * linear_func(x)
    return value % 8
```

Each candidate polynomial was tested against all computational basis states. Only exact matches modulo 8 were accepted as valid representations of the unitary.

Once a valid polynomial was found, it was translated into an OpenQASM circuit by computing each linear function with CNOT gates, applying the corresponding T or $T^\dagger$ gate, and uncomputing.

Calculation Example

Let us denote the four-qubit computational basis state by $|x_0x_1x_2x_3\rangle$. The action of U on this state can be expressed as:

$$U|x_0x_1x_2x_3\rangle = e^{i\pi \sum_j 7f_j(x_0,x_1,x_2,x_3)/4}|x_0x_1x_2x_3\rangle,$$

where each f_j is a parity function over a subset of the qubits. Evaluating modulo 8, we have:

$$7f_j \equiv -1 \pmod 8.$$

Therefore, each $T^\dagger$ gate effectively contributes a phase of $-\pi/4$, and the total phase for a basis state is obtained by summing all contributions:

$$\text{Phase}(x_0, x_1, x_2, x_3) = \sum_j \left(-\frac{\pi}{4} f_j(x_0, x_1, x_2, x_3)\right).$$

11.6 Results and Analysis

The best solution obtained using this approach had:

$$\text{T-count} = 4,$$

and an operator norm distance from the target unitary of

$$\|U_{\text{synth}} - U_{\text{target}}\| = 0.7653668647301796.$$

While this solution was not optimal, it is still meaningful for several reasons:

- The circuit exactly captures the dominant phase structure of the unitary.
- The T-count is relatively low compared to naive CCZ-based constructions.
- The remaining error indicates that the phase polynomial matches the target only approximately, highlighting the limitations of the restricted search space.

12 Exploring Multi-Objective Quantum Compilation via Pareto Efficiency

To systematically explore the space of compiled quantum circuits, we developed a *solution explorer* that leverages the concept of Pareto efficiency. Our goal was to generate a variety of circuit solutions for a given unitary and identify trade-offs between multiple cost metrics, including T-count, circuit depth, CNOT count, and the number of ancilla qubits.

12.1 Circuit Representation and Metrics

Each candidate circuit solution was represented as a structured object containing its OpenQASM code, T-count, depth, CNOT count, number of ancilla qubits, total gates, and a brief description of the compilation strategy. We considered the following metrics as objectives for optimization:

- **T-count:** Number of T gates, which are costly in fault-tolerant architectures due to magic state preparation.
- **Circuit depth:** Maximum number of sequential layers of gates.
- **CNOT count:** Number of entangling operations, significant for hardware error rates.
- **Ancilla qubits:** Extra qubits used to facilitate certain compilation strategies.

These metrics were selected because they directly influence both the feasibility and the efficiency of executing the circuit on real quantum hardware.

12.2 Pareto Efficiency in Solution Exploration

A solution is considered *Pareto efficient* if no other solution improves at least one metric without worsening another. Formally, a candidate solution S_i dominates S_j if, for all objectives k, $S_i[k] \leq S_j[k]$, and for at least one objective, $S_i[k] < S_j[k]$. Our program evaluated dominance across all candidate solutions to identify the *Pareto frontier*, i.e., the set of non-dominated solutions.

This approach allowed us to visualize the trade-offs inherent in quantum compilation: some solutions minimized T-count at the expense of increased depth or CNOTs, while others prioritized low depth or minimal qubit usage. By highlighting these non-dominated solutions, the Pareto frontier guided the selection of candidate circuits for further analysis and hardware-specific optimization.

12.3 Implementation of the Solution Explorer

Our multi-objective compiler generated several strategies for each unitary:

1. **Minimizing T-count:** Allowed ancilla qubits and leveraged XOR-based Reed-Muller decomposition.
2. **No ancilla:** In-place computation strategies, accepting higher T-count.
3. **Minimizing depth:** Parallelization of independent gates.
4. **Minimizing CNOT count:** Reduced entangling operations where possible.
5. **Balanced trade-off:** Attempted moderate values across all metrics.

After generating these solutions, the program computed the Pareto frontier and produced a *solution menu* that recommended circuits based on metric-specific optimization and hardware suitability. Each hardware profile incorporated relative costs for T gates, CNOT gates, ancilla qubits, and circuit depth, allowing us to score solutions according to platform-specific constraints.

12.4 Observations and Limitations

While Pareto efficiency provided a structured way to explore solution space, it had several practical limitations:

- **Limited resolution:** With only a few candidate strategies, the Pareto frontier could be sparse, leaving large gaps in the trade-off landscape.
- **Lack of prioritization:** Pareto optimality treats all objectives equally; in practice, some metrics, such as T-count, may dominate hardware cost considerations.
- **Scalability:** As the number of objectives and candidate solutions grows, computing dominance becomes increasingly expensive.

Despite these limitations, the Pareto framework was valuable for providing *insight into trade-offs*, guiding strategy selection, and informing hardware-aware compilation choices. It enabled us to construct a *solution explorer* that systematically evaluated and visualized competing objectives, supporting informed decisions in quantum circuit optimization.

13 Conclusion

In this work, we explored a broad range of fault-tolerant quantum compilation problems, spanning exact Clifford constructions, approximate single-qubit rotation synthesis, Hamiltonian simulation, state preparation, and general two-qubit unitary approximation. Rather than pursuing a single universal compilation strategy, our results highlight that effective fault-tolerant design is inherently problem-dependent and strongly shaped by the underlying structure of the target operation.

Across multiple challenges, identifying hidden algebraic or physical structure—such as parity-phase decompositions, basis-change identities, Fourier structure, or phase-polynomial representations—proved to be the most powerful tool for reducing circuit complexity. When such structure was available, exact or near-exact implementations with minimal T-count were often achievable. In contrast, problems lacking clear structural signatures resisted direct synthesis and instead required heuristic or numerical approaches, exposing the limits of brute-force and combinatorial search in high-dimensional unitary spaces.

A recurring theme throughout our experiments was the fundamental tradeoff between accuracy and resource cost. High-fidelity Clifford+T synthesis of irrational-angle rotations consistently led to extremely large circuits dominated by T-gates, while approximate or constrained methods achieved dramatic reductions in gate count at the cost of controlled fidelity loss. This tension was especially apparent in state preparation and random unitary approximation, where simple randomized or resource-constrained strategies delivered surprisingly competitive performance relative to much more expensive exact constructions.

Overall, our results reinforce a central lesson of fault-tolerant quantum computing: compilation is not merely a mechanical translation from unitaries to gates, but a design problem that requires balancing mathematical structure, numerical optimization, and physical resource constraints. As fault-tolerant architectures mature, practical quantum algorithms will increasingly depend on hybrid approaches that combine analytical insight with adaptive approximation, rather than relying on any single notion of optimality.